UNCLASSIFIED

Technical Report
CMU/SEI-91-TR-29
ESD-TR-91-29
October 1991

A Critical Review of the Current State of IPSE Technology

Alan W. Brown
Software Development Environments Project

586436

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

SEI Joint Program Office
ESD/AVS
Hanscom AFB, MA 01731

Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER

[illegible]
Charles J. Ryan, Major, USAF
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense.
This report was funded by the U.S. Department of Defense.

Contents

Chapter 1

Introduction

"We shall not cease from exploration
And the end of all our exploring
Will be to arrive where we started
And know the place for the first time."

T.S. Eliot, *Little Gidding.*

Abstract: In the past ten years there has been a great deal of interest in the concept of an Integrated Project Support Environment (IPSE) as a complete, unifying framework of services supporting most (or all) phases of software development and maintenance. In this paper we evaluate the current state of research work in this area, suggest some reasons for the relative lack of success, and make proposals for ensuring measured progress in the future.

The concept of an Integrated Project Support Environment (IPSE)[1] was developed in the late 1970's and early 1980's in response to a recognition that the traditional notion of a set of program development tools (editor, compiler, debugger) is only a small component of a much larger set of facilities required to coordinate the many tasks involved in large-scale software production. While various descriptions of many of the concepts are available, in many ways the seminal work on IPSEs and IPSE architecture is the Stoneman report[6, 7] defining the architecture of an Ada Programming Support Environment (APSE). Attempts at defining an architecture for a *project* support environment have been greatly influenced by this work. In that report a basic

[1] Also variously called a Software Development Environment (SDE), Software Engineering Environment (SEE), and Integrated Software Factory (ISF).

"onion layer" architecture was defined with kernel support at the center (operating system and database services), a minimal toolset surrounding that kernel (essential support tools such as an editor, compiler and debugger), and an outer layer of further tools (specific tools to tailor the environment to different software development techniques). While decentralized approaches are now being examined, the Stoneman approach has been the touchstone for much subsequent work.

It has now been more than 10 years since Stoneman. It is high time that we stood back and evaluated the current situation with regard to IPSEs and IPSE technology, the lessons learned, the progress made during the 1980's, and the likely directions for the 1990's. Among the questions we should ask are:

- How much progress has been made since Stoneman?
- What are our experiences after this time?
- What lessons have we learned about good and bad approaches to IPSEs?
- Where is much of the current research effort being applied?
- What do we see as the key issues for the future?

In addressing these questions we draw together a number of important issues that are currently the topic of much discussion both in Europe and the US concerning the likely direction of work in the field of computer support for software development. Hence, we hope to point the way forward for future work in this important area of software engineering.

We begin by examining some of the reasons for the relatively slow take up of IPSE technology in the commercial world, then turn our attention to alternative solutions which have seen greater success. We then briefly highlight the main areas of IPSE research currently being pursued, before concluding with an analysis of the key issues for the future.

1.1 What is an IPSE?

For the purposes of this paper, we distinguish two classes of approach towards computer-based software development support, based on the definitions given in [21].

The first class, which we refer to as *Integrated Project Support Environments (IPSEs)*, can be seen as the direct descendents of the Stoneman work. The primary aim of these systems is to investigate environment infrastructure technology by concentrating on the definition of an environment framework. The majority of the work in this area has explored the common services that need to be provided by such a framework, and the consequent interaction between the framework and tools which are embedded within it.

Two concepts are of particular interest here:

- We can distinguish between the IPSE as a framework of common services, and a "populated IPSE" (i.e. the IPSE together with its embedded tools). One IPSE framework may be populated with different sets of tools to suit different software development needs.

- The IPSE framework (implicitly or explicitly) supports some notion of a software development process. By the very nature of its architecture, the IPSE will make some models of the software development process easier or more difficult to support. In some cases, a single, well-defined software process is built into the IPSE framework from the outset. In other cases there is more flexibility over the development process supported.

The second class is the *Computer-Aided Software Engineering (CASE) tool* approach. Here, driven by market needs, simple software development tools have been enhanced to provide more comprehensive services for data management, better user interface facilities, and increased tool functionality.

There are two notable trends in the CASE tool world:

- In recent years the range and availability of tools has increased greatly. Tools now cover some aspects of multiple stages of software development, from requirements elicitation through to system maintenance.

- As users have gained experience with CASE tools they have seen the need for combining individual tools to cover more (or all) of the software development life-cycle. Two approaches to integration of CASE tools have been attempted — *ad hoc* integration through filters, pipes, and transfer protocols, and "vendor pacts", where CASE tool vendors share knowledge of their tools to allow them to work together in the hope of increasing their market share. The term "integrated CASE (iCASE)" is sometimes used in any, or all, of these situations.

In the remainder of this paper we concentrate on the use of IPSE technology, the reasons for its relative lack of success, how it compares to the more commercially active CASE market, and how future research should focus on a number of key areas in order to ensure maximum leverage from the large body of expertise and knowledge gained from existing work in IPSE technology.

Chapter 2

Reasons for the Lack of IPSE Use

It is not being overly dramatic to say that IPSEs and IPSE technology are not currently being used in anywhere near the number of commercial organizations that were envisaged at the start of the 1980's. What is more, the development and acceptance of IPSE technology has taken much longer than was expected. For example, the vision of the UK Government-funded Alvey program in the early 1980's was that the Information System Factory (ISF), employing knowledge-based techniques in the support of all aspects of computer system development, would be in place within the first few years of the 1990's. Clearly, this will not be the case.

2.1 IPSE Achievements

It is wrong to believe, however, that no progress has been made at all. Significant achievements have been recognized through the IPSE work that has taken place over the last decade. Here we choose to highlight two of those achievements:

- A major focus of IPSE research has been in the area of data management support for the large, complex set of data items that typically must be recorded and controlled[3, 1]. Following initial attempts at using commercially available database systems as IPSE repositories, the problems encountered have resulted in detailed examinations of IPSE data management requirements, leading to enormous activity in the database world to address those requirements. This has produced definitions of new IPSE data models[9], initiated development of the field of Object Management Systems (OMSs) for IPSEs[12], and provided a major catalyst for work in the area of object-oriented databases (OODBs)[4]. Indeed, support for computer-aided design (CAD) in general, with IPSEs as a particular class of CAD system, has been at the very heart of the current revolution in database technology.

- The main aim of an IPSE is to represent and support some notion of a software development process. Initially, the process supported was implicitly embedded in the IPSE through the choice of available tools, and the restrictions placed on their interconnection. Much of the IPSE work in the past few years, however, has been in attempting to make process notions more explicit, user-definable, and executable[19, 18]. Understanding, modeling, and automating the software development process is also currently a very active area of software engineering research in general[14]. This work, while motivated by many factors, has both drawn from, and fed into the work on IPSEs. Certainly, the insights gained through IPSE work have been of great benefit to this field, improving our understanding of the software development process and how to support it.

2.2 IPSE Shortcomings

Based on reported IPSE user experiences and our knowledge of the IPSE field gained from talking with fellow IPSE researchers, end users looking for suitable IPSE and CASE technology, and third party tool developers, we can see a number of reasons why IPSEs and IPSE technology are not in widespread use. The most important of these reasons appear to be:

1. *Little documented evidence of the cost effectiveness of IPSE technology.*

   There have been successful examples of the use of IPSE technology. However, there is very little publicly available and reliable data describing what technology was used, how it was chosen, what problems it addressed, and what measurable benefits were obtained. There are obvious reasons why—commercial confidentiality, lack of time to write up results, reluctance to write about failures as well as successes, and perhaps, a lack of fora for discussing "real" user experiences. However, one of the major problems in this area is both the lack of suitable data on which to perform such analyses, and the lack of suitable meters with which to measure changes in productivity. Figures such as "a 12-fold increase in productivity" can often lead to long debate on what is being measured, how it is being measured, and what the new values are being compared to. For example, with the SAFRA report[17], it is tempting to conclude that the large gains in productivity that are claimed are more a result of changes in working practices and general software process improvements that accompanied the introduction of the IPSE technology than a consequence of the IPSE technology itself. This is not to say that the IPSE technology was without value; rather that it constitutes only part of a larger change in culture.

   It may be difficult to document the experiences of using IPSEs, but without a body of evidence to back up the case for IPSE technology many commercial companies will be unable to justify making an investment in the technology.

2. *The size of IPSEs.*

   In general, an IPSE is attempting to manage a large, complex set of problems in software development. This leads to IPSEs themselves being large, complex software systems. There are important consequences of this:

   - They are inevitably costly to purchase, maintain, and use, with possibly a large investment required in new hardware on which to run the IPSE, training of staff, consultancy fees, and upgrades to the system as new releases of the IPSE infrastructure and its tools are produced.

- Their size, complexity, and relative newness can result in instability in IPSE operation, manifesting itself in poor design, slow performance, low availability due to system crashes, and so on. Users may often find themselves in the position of being one of only a small number of customers using that IPSE, with the uncomfortable feeling that they are debugging the IPSE for its developers.
- A complex piece of system software such as an IPSE requires technical support staff to monitor, tune, and adjust its operational parameters increasing the cost of using the IPSE.
- There will inevitably be some effect on the software development processes employed by an organization through the introduction of this new technology. Just the time needed to become familiar with an IPSE may mean that initially productivity actually decreases before (hopefully) increasing above previous levels. More often, however, the introduction of IPSE technology is seen as part of a larger program in software process improvement[14], when it is all too easy to either expect too much of the technology supporting the process changes, and to blame the technology when the changes do not keep to plan.

3. *Lack of flexibility.*

   The ideal of providing a tailorable, configurable IPSE which is customized as necessary for different organizations, projects, and individuals is in reality a long way from current practice. The more usual situation is that the IPSE provides services through a fixed set of tools, with at best the possibility of new tools being added by referring back to the IPSE builders (e.g. ISTAR[8] and ECLIPSE[2]). Such inflexibility is a great drawback to the use of IPSEs across different commercial organizations in the different development processes they employ, as well as within a single commercial organization which employs different development processes for different projects.

4. *Concern about long-term investment.*

   The IPSE market is very volatile at present, with new ideas, approaches, mechanisms, and techniques being announced quite regularly. For commercial organizations contemplating using IPSE technology, they are naturally greatly concerned that they make the "right" choices. This is hampered at present by confusion surrounding the technology and its advantages and disadvantages. A major aim of much recent work has been to help with the necessary analysis and comparison of the many possible approaches, techniques and mechanisms (for example, the European Computer Manufacturers Association (ECMA) reference model for Computer-Assisted Software Engineering Environments[10]).

   Perhaps the greatest problem in this area involves the issue of standards. As is often the case, the idea of standards is attractive to commercial organizations (both vendors and users) as it leads to thoughts of portability, compatibility, larger market share, extended product life-time, and so on. Unfortunately, in the IPSE world there is still much debate over which areas of IPSE technology should be standardized, whether the time is right for standardization in those areas, which existing standards are most appropriate, and what new standards should look like. Many commercial organizations would prefer that some of these issues were closer to being resolved before making their decisions about what technology to use. Indeed, there is also the argument that the best standards are those that evolve naturally through a process of "natural selection". However, this is only possible when a range of competing systems are used over an extensive period of time which is not yet the case for IPSEs.

In summary, we recall that the main focus of IPSEs is the infrastructure in which the development tools can be embedded. Through the definition of appropriate services within the infrastructure, IPSEs hope to provide increased tool portability, improved tool integration mechanisms, and a single approach to common development activities. However, in practice we see that these advantages can often be nullified by a series of pragmatic issues. We have highlighted a few of these, including the strategic nature of the purchase of IPSE technology, the high cost of purchase, the probable changes required to existing working practices, and the lack of documented evidence of successful projects.

One more point to note is that IPSEs are often purchased by software managers for use by end user developers. If this process is badly managed, there will inevitably be end user resentment and frustration at the imposition of the new technology. This can only be overcome by ensuring that the IPSE is introduced as *supporting* the existing development processes, and with the cooperation and understanding of the eventual end users.

Chapter 3

Alternative Solutions

While the introduction of IPSE technology has been very limited in scope, the purchase and use of individual CASE tools has been widespread. Many commercial organizations have experimented with automated software development tools. While initially these tools were aimed at the implementation phases of software development (coding, debugging, version control), more recently there has been rapid development of tools addressing other aspects of software development. For example, there are now many tools which address some aspects of system design, data modeling, and management support. Such tools are typically classed together under the often-used term "CASE tool". We can refer to commercial organizations that use CASE tools in this way as "first generation" CASE tool users.

As the use of such tools has grown, there has been increasing interest in ways of combining CASE tools to support more of the software life-cycle. This "second generation" of CASE tool users has experienced the advantages of individual tools, but now requires tools which work in combination. What many commercial organizations are doing is buying collections of CASE tools and "glueing" them together in the best way they can, rather than move to an IPSE solution. The "glue" is provided through a common set of data definitions used between the tools, use of a common data transfer protocol, or through the writing of individual conversion routines to link the tools used by the organization. The term "integrated CASE (iCASE)" is often used in this context. There are, of course, many obvious disadvantages to this inflexible, piecemeal approach to tool integration. However, the popularity of this approach has a very pragmatic rationale:

- It provides what is often seen as a relatively low cost approach to integrated tool support (although in practice this cost can be high). One of the main drawbacks of IPSE technology is its high cost. Following a CASE tool approach provides not only a low cost entry point into the technology, but also the possibility of developing the CASE tool environment through incremental investment in new tools.

- The investment that is made is seen as being easier to justify. Most of these commercial organizations have already made some investment in CASE tool technology. This route to integrated support appears to be more incremental, and is acceptable because, in general, the approach employs simple, well understood technology with much less demand on hardware and administrative costs. Also, the range of CASE tools and CASE tool vendors means that there is more competition for business, and less reliance on a single supplier.

- With IPSE solutions the technology is complex, and in many cases the users of the system have a very limited understanding of how it works. This can lead to resentment, frustration, and a poor attitude to its use. In contrast, CASE tools are often (in principle) deceptively simple. The technology is not seen as "getting in the way" as they are easier to understand, manage, and control.[1] The overall feeling is that the purchasing organization is "more in control" of the processes of purchasing, developing, and using the environment.
- For most software systems the end-user organization is also responsible for systems administration, and, in particular, the task of systems integration. This is seen in two ways. First, from a technical view, the tool must be integrated with other tools that are in use. At its most basic this could be ensuring the tools run on the same operating system, or use the same window manager. For CASE tools the integration technology, while providing only low levels of tool integration, involves skills that are more readily available to the end-user organization (e.g. UNIX expertise). For IPSEs, the complex infrastructure services often make tool integration a much more specialist task. Second, from a process view, the tool must be integrated into the organization's software development procedures. The piecemeal approach provided by individual CASE tools can offer a great deal of flexibility in this regard. In IPSEs the process definition mechanisms are often either fixed to a single pre-determined approach, or are sufficiently complex to require specialist support to make use of them.
- In some ways many of the CASE tools can be seen as rather unambitious in their aims. However, they are seen as tackling the obvious and most pressing needs of many commercial organizations — version control, documentation support, and structured design method support. It may well be a situation in which the organizations must learn to walk before they attempt to run! The work by Watts Humphrey and others[14] defining a series of levels of understanding of the software development process within commercial organizations certainly supports this point. The vast majority of the organizations examined were seen to be at the lowest level of understanding, or "maturity".

  This leads to two schools of thought with respect to IPSE technology — either the introduction of complex technology to support a poorly understood software development process will rarely prove to be of benefit, leading instead to support for an *automated*, ill-defined software development process, or the more process-mature organizations do not necessarily require any more sophisticated development tools than the less mature ones, as they are able to make better use of those tools in a development process that is monitored, controlled, measured, and repeatable. Both of these cases emphasize the secondary role of IPSE technology.

  In this situation, perhaps simple solutions taken in small, well defined steps are the most effective. It is certainly difficult to believe that revolutionary steps such as the purchase of an IPSE will, *in themselves*, be effective.

[1]As users of some of the large CASE tools will know, the perception of ease of management and use of CASE tools is not always borne out in practice!

UNCLASSIFIED

In summary, we note that on most occasions the purchase of a CASE tool is *not* seen as a strategic decision, but more as a pragmatic one. For example, it is often found to be easier to obtain money and management support for purchasing a new CASE tool, or integrating a set of CASE tools, than for investing in IPSE technology. This is due to the incremental nature of the investment, the more visible improvements in productivity they often bring, and the more manageable complexity of the new software.

Chapter 4

Where is All the Money Going?

It can certainly not be claimed that the lack of progress in introducing IPSE technology to industry is due to a lack of investment in the technology. In fact, the amounts of money being made available to work in this area are quite astounding. Examples of past and current expenditure on IPSE technology development may be illustrated with examples from the Government sponsored research taking place in Europe and the USA:[1]

- The UK Government funded Alvey program part-funded three parallel IPSE projects — Aspect, ECLIPSE, and IPSE2.5. The total investment was in excess of 20 million pounds. The focus of this work was primarily on the data integration technology required for an IPSE, investigating the use of file based, database, and knowledge based system technology as the basis of an IPSE.

- The European Commission has funded, or part-funded, a number of IPSE related projects under its ESPRIT programs, many of which are on-going. Examples include the 400 million dollar Eureka Software Factory (ESF) project, the estimated 600 man-years of effort on Atmosphere, and 200 man-years on ARISE. In all three of these projects the work taking place involves large consortia of different industrial/academic institutions, in a number of countries. As a result, the work itself tends to be very diverse in nature. However, one of the main foci of the much of the work is a concentration on process support mechanisms.

- In the U.S. there have been a number of government related initiatives. Notable amongst them are the Ada Programming Support Environment (APSE) work (individual projects sponsored by the Army and Navy) at a total cost of approximately 100 million dollars and the software development environment work carried out as part of the DARPA funded STARS program estimated at 75 million dollars. While the APSE work was focused directly on support for Ada software development, the STARS project has very extensive goals, aiming to develop a large, generic frameworks suitable for supporting a wide range of computer system developments.

[1]One point to note is the scale of some of these projects, involving a large group of people, from many companies and universities, often from different countries. This leads to communication difficulties, coordination problems, and inevitable political manoeuvrings within the projects.

A target of much of the European research effort over the past ten years has been the work on the Portable Common Tools Environment (PCTE) and its related projects, and in the U.S. on its counterpart, the Common APSE Interface Set (CAIS), and its derivatives. These large initiatives have attracted (and continue to attract) much attention and a great deal of the research effort and funds.

However, given the earlier discussion on the reasons for a lack of widespread use of IPSE technology, we are inevitably drawn to the conclusion that the emphasis of much of the current IPSE research is misguided. In particular, we would suggest that IPSE research in the following areas would be useful:

- *Smaller, user-oriented, application-led projects.*

  This would enable user requirements to be more clearly identified, and specific solutions examined in detail.

- *Gaining extensive experience with existing tools, toolsets, and IPSEs.*

  This would not only increase awareness of existing good practices, but would also allow us to collect data on productivity and quality using existing approaches, and enable further work to take place in developing suitable metrics for this field.

- *Evolutionary approaches rather than revolutionary.*

  It is important to realize that the acceptance and use of IPSE technology may be as much based on pragmatic issues as on technical ones, with the result that the more intellectually appealing, and technically demanding, solutions are not necessarily those that will be most beneficial to users in practice.

- *Widening the appeal of work on IPSEs.*

  At present there is an ideological split between those working in the the area of real-time, embedded systems such as avionics, and those in the data processing and information systems world. In the past IPSEs have been targeted more towards the former, rather than the latter market. It is clear that there is much to learn from the commercial CASE tool world, with the hope that both markets will be served by the resultant architectures, methods, and tools. Building on the experiences of both these fields could provide the basis we require.

Chapter 5

Key Issues for the Future

Having highlighted some of the reasons for the lack of widespread IPSE use, we now look at what we believe will be major issues for the future in attempting to resolve the current situation. While there are clearly many issues on which we could focus, in this discussion we choose to address some of the issues that we believe are not currently seen as being crucial to future progress, but that we forecast will be of vital importance.

- *A return to the IPSE users' requirements.*

  There remains a certain lack of focus in current IPSE work which we believe can be traced directly to a lack of understanding of the different IPSE users and their requirements. We are now at an appropriate point in IPSE understanding at which to return to the potential IPSE users to discover why they are not adopting the technology currently available. There certainly appears to be concern from those skeptical of the value of IPSE research that the work has concentrated too much on the technology *per se*, without sufficient regard for the areas in which the most leverage can be obtained from using such technology.

  In particular, it appears that the gap between IPSE researchers and potential IPSE users is growing. Some people believe, for example, that the large, Government-funded, collaborative research projects are in danger of becoming too far out of step with user needs[13]. Similarly, this issue was also a major focus of a recent workshop looking at the possible convergence of the PCTE and CAIS proposals[15], with some participants concluding that there was a real need for much more work to establish concrete, low level end user requirements before the work could progress.

  A second issue is that the level of requirements definition in the past has often assumed particular technical solutions. For example, both the Requirements and Criteria (RAC) document[16] for the Common APSE Interface Set (CAIS), and its European counterpart[11] (EURAC) for the Portable Common Tools Environment (PCTE) were produced *after* their respective system definitions had been produced, and were an attempt to influence that definition's future direction. At a more fundamental level, it remains be established that the technical solutions provided really do meet the end users' needs.

Running counter to this, of course, is the argument that users often do not know or fully understand their requirements in this area. This highlights the need for an intelligent approach to requirements elicitation and analysis, and for the incremental development and introduction of IPSE technology into an organization. Hence, it is the role of the IPSE designers and researchers to ensure that their vision for the future is tempered by the IPSE user's current reality.

- *An understanding of the process as a precursor to the introduction of solutions.*

  This quite obvious point is very often overlooked. However, it must be emphasized that IPSE technology is not a replacement for well defined, visible software development procedures – it merely supports those procedures. As Humphrey's work on software conformance levels[14] has shown, the level of understanding of software processes within many commercial organizations is particularly low. While an IPSE can help support a process when it is in place, it is not a substitute for sensible development procedures. Many companies are beginning to recognize this, and a number of companies are actively involved in defining and administering so-called "process improvement programs". The result of this work may not only be companies who better understand the way in which they develop software, it may also lead to a better understanding of the software development process itself. This will feed directly into the process modeling aspects of IPSE technology.

- *Useful models for understanding IPSE technology.*

  One of the main barriers to greater acceptance of IPSEs is the confusion and misunderstanding surrounding the whole area. Part of the reason for this is that we have not had sufficiently detailed, yet generally applicable models of IPSEs with which to explain, reason, understand, and teach. The Stoneman model, useful as it was at the time, provides too coarse a model for any detailed work. In addition, the models that have been defined (such as the Stoneman model) have IPSE developers as their primary audience. Other key users (such as tool integrators and IPSE end users) have to a large extent been ignored.

  Very recently this problem has started to be addressed in a number of ways. In particular, we can highlight two approaches. First, a reference model for analyzing and examining existing and proposed IPSE solutions has been proposed and accepted by the European Computer Manufacturing Association (ECMA)as an approved technical report[10]. This provides us with a *mechanistic* view of the current systems. Second, recent work [22, 20, 5, 21] has looked at the key issue of integration within an IPSE and proposed measures for different aspects of the integration. This provides a *semantic* view of the integration requirements of an IPSE system. These approaches are complementary, and both are useful and helpful in providing us with a better understanding of the complex issues involved in this field.

  However, there is one further view of integration which has yet to receive significant attention — a *process* view. This third approach addresses the integration of tools within an organization's existing software development process. No generally applicable models are currently available in this area.

- *An evolution towards federated architectures.*

  In characterizing work on computer-based support for software development we see that while IPSE technology has concentrated on the framework services, CASE environments have emphasized the individual tools. There are obvious grounds for believing that future systems must employ the best of both approaches. One recent proposal[21] examines the ideas of a "federated" architecture based on process support through the concept of environment services. In this way, the distinction between a "tool" and a "framework service" is dissolved. Instead, users request services that are provided by the environment. Details of tools and framework are hidden below. While more work is clearly required to fully validate such an approach, it does seem an attractive way to ensure that the strengths of both IPSE and CASE technology are preserved in future systems.

In summary, we see that a need to better understand IPSE user requirements remains, ideally before embarking on further large, expensive system developments. Arguably, we are still not in a position to make such a complete and detailed analysis of IPSE user requirements. However, as happens in the development of any large, complex system, requirements analysis and design proceed hand-in-hand, with the results of partial designs and implementations feeding directly into further requirements elicitation and analysis activities. This leads us to believe that requirements analysis for IPSEs, together with the design and development of IPSE mechanisms, must take place in parallel, using the requirements to evaluate the designs, and the designs as the basis for elicitation of further IPSE requirements. Approaches aimed at combining the strengths of both IPSE and CASE technology seem an attractive way forward.

Chapter 6

Summary

In this paper we have tried to summarize some of the main issues currently of concern to IPSE workers by looking towards the future of IPSEs and IPSE technology and identifying some of the likely key areas.

IPSE research has been successful in a number of important ways. In particular, there have been great advances in understanding aspects of the technology required to support software development. This has had repercussions in other areas of software engineering. Interest in object-oriented databases, for example, has been spurred on by the evolving knowledge of the necessary data support mechanisms for an IPSE[4].

However, while it is clear that there have been great advances since the seminal work of the early 1980's described in the Stoneman report, there is still a long way to go. Our main hope is that the relative lack of success does not obscure the many great advances in understanding that have been made in the past decade. In particular, we should see the main success of the work as helping us to re-focus our attention on the issues which are important for the future — understanding the nature and requirements of IPSE users, the importance of a knowledge of the development process to software production, and an emphasis on getting the best out of available technology to improve future development. The proposals made in this paper have been constructed with the aim of ensuring that future work builds on these successes.

The 1990's should prove an interesting and exciting time for IPSE workers, with many major issues still to be tackled. Perhaps after all this time, following a great deal of exploration, we are only just beginning to understand some of the issues that must be addressed in this work, and the important questions that remain to be answered. In fact, we can now perhaps claim no more than to *"know the place for the first time"*.

Acknowledgements

Many of the ideas reported in this paper have benefited from discussions with John McDermid at the University of York, and Anthony Earl at Hewlett-Packard Labs in Bristol.

Earlier drafts of this paper were improved through the advice of colleagues at the SEI, notably Larry Druffel, Bob Ellison, Peter Feiler, Dennis Smith, and Kurt Wallnau, and following comments from Tricia Oberndorf of NADC. Dick Martin at the SEI helped with some of the figures quoted.

My thanks also to Peter Hitchcock for the T.S. Eliot quotation!

Bibliography

[1] P.A. Bernstein. Database System Support for Software Engineering. *Proceedings of the 9th International Conference on Software Engineering*, pages 166–179, March 1987.

[2] M.F. Bott, editor. *ECLIPSE – An Integrated Project Support Environment.* Peter Peregrinus, 1989.

[3] A.W. Brown. *Database Support for Software Engineering.* Chapman and Hall, 1990.

[4] A.W. Brown. *Object-Oriented Databases and their Application to Software Engineering.* McGraw-Hill, 1991.

[5] A.W. Brown and J.A. McDermid. On Integration and Reuse in a Software Development Environment. In Fred Long and Mike Tedd, editors, *Software Engineering Environments '91.* Ellis Horwood, 1991.

[6] J.N. Buxton. *Requirements for APSE – Stoneman.* U.S. Department of Defence, February 1980.

[7] J.N. Buxton and L.E. Druffel. Requirements for an Ada Programming Support Environment: Rationale for Stoneman. In *Proceedings of COMPSAC 80*, pages 66–72. IEEE Computer Society Press, October 1980.

[8] M. Dowson. ISTAR – An Integrated Project Support Environment. *Proceedings of 2nd SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments*, pages 27–33, December 1986.

[9] A.N. Earl and R.P. Whittington. Capturing the Semantics of an IPSE Database – Problems, Solutions and an Example. *Data Processing*, 27(9), November 1985.

[10] ECMA. A Reference Model for Computer-Assisted Software Engineering Environments. *ECMA Report Number TR/55*, January 1991.

[11] GIE Emeraude, Selenia, and Software Sciences. Requirements and Design Criteria for Tool Interfaces (EURAC), 1987.

[12] F. Gallo, R. Minot, and I. Thomas. The Object Management System of PCTE as a Software Engineering Database Management System. *Proceedings of 2nd SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments*, pages 12–15, December 1986.

[13] A. Gilles. Conference Report. *Information and Software Technology*, 33(2), March 1991.

[14] W.S. Humphrey. *Managing the Software Process.* Addison-Wesley, 1989.

[15] Yard Ltd. *Proceedings of the PCIS Workshop.* June 1991.

[16] US Department of Defense. Requirements and Design Criteria for the Common APSE Interface Set, 1986.

[17] C. Price. *SAFRA - A Debrief Report.* NCC Publications, 1987.

[18] R.A. Snowdon. A Brief Overview of the IPSE2.5 Project. *Ada User*, 9(4):156–161, 1988.

[19] R.N. Taylor and Others. Foundations for the arcadia environment architecture. *ACM SIGPLAN Notices*, 24(2):1–13, February 1989.

[20] I. Thomas and B. Nejmah. Tool Integration in a Software Engineering Environments. Technical Report SESD-91-11 Revision 1.1, Hewlett-Packard, June 1991.

[21] K.C. Wallnau and P.H. Feiler. Evolving Towards Federated, Services-Based CASE Environments. *Submitted for publication*, June 1991.

[22] A. Wasserman. Tool Integration in Software Engineering Environments. In F. Long, editor, *Software Engineering Environments*, number 467 in Lecture Notes in Computer Science, pages 138–150. Springer-Verlag, 1990.

UNCLASSIFIED

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S CMU/SEI-91-TR-29	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESD-91-TR-29
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute 6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office
6c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213	7b. ADDRESS (City, State and ZIP Code) ESD/AVS Hanscom Air Force Base, MA 01731
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office 8b. OFFICE SYMBOL (if applicable) ESD/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003
8c. ADDRESS (City, State and ZIP Code) Carnegie Mellon University Pittsburgh PA 15213	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO: 63756E PROJECT NO.: N/A TASK NO: N/A WORK UNIT NO.: N/A

11. TITLE (Include Security Classification)
A Critical Review of the Current State of IPSE Technology

12. PERSONAL AUTHOR(S)
Alan W. Brown

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM TO	October 1991	22 pp.

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse of necessary and identify by block number)
IPSE
Integrated Project Support Environment
Software Environments Environments

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In the past ten years, there has been a great deal of interest in the concept of an Integrated Project Support Environment (IPSE) as a complete, unifying framework of services supporting most (or all) phases of software development and maintenance. In this report, we evaluate the current state of research work in this area, suggest some reasons for the relative lack of success, and make proposals for ensuring measured progress in the future.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED SAME AS RPTDTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution
22a. NAME OF RESPONSIBLE INDIVIDUAL Charles J. Ryan, Major, USAF	22b. TELEPHONE NUMBER (Include Area Code) (412) 268-7631 22c. OFFICE SYMBOL ESD/AVS (SEI)

ABSTRACT —continued from page one, block 19